Multimed Tools Appl
https://doi.org/10.1007/s11042-017-5326-1

Feature learning for steganalysis using convolutional neural networks

Yinlong Qian[1,2] · Jing Dong[2] · Wei Wang[2,3] · Tieniu Tan[2]

Received: 19 December 2016 / Revised: 17 September 2017 / Accepted: 19 October 2017

Abstract Traditional steganalysis methods usually rely on handcrafted features. However, with the rapid development of advanced steganography, manual design of complex features has become increasingly difficult. In this paper, we propose a new paradigm for steganalysis based on the concept of feature learning. In our method, Convolutional Neural Network (CNN) is used to automatically learn features for steganalysis. To make CNN work better for steganalysis, we incorporate domain knowledge of steganalysis (i.e. enhancing stego noise and exploiting nearby dependencies) when designing the CNN architectures. We further propose to use model combination to boost the performance of CNN based method. Additionally, a cropping strategy is proposed to enable the CNN based model to deal with arbitrary input image sizes. We demonstrate the effectiveness of the proposed method against state-of-the-art spatial domain steganographic algorithms such as HUGO, WOW, S-UNIWARD, MiPOD, and HILL-CMD. To help understand the learned features from CNN, we provide visualizations of the learned filters and feature maps. Finally, we also provide quantitative analysis of the learned features from convolutional layers.

✉ Jing Dong
jdong@nlpr.ia.ac.cn

Yinlong Qian
ylqian@mail.ustc.edu.cn

Wei Wang
wwang@nlpr.ia.ac.cn

Tieniu Tan
tnt@nlpr.ia.ac.cn

[1] Department of Automation, University of Science and Technology of China, 96 JinZhai Road, 230026, Hefei, China

[2] Center for Research on Intelligent Perception and Computing, Institute of Automation, Chinese Academy of Sciences, 95 Zhongguancun East Road, 100190, Beijing, China

[3] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing, 100878, China

Published online: 20 November 2017

Keywords Steganalysis · Steganography · Deep learning · Feature learning · Convolutional neural networks

1 Introduction

The aim of steganalysis is to detect the presence of secret messages in digital media such as digital image, text, audio, video, etc. Because of the popularity of images in digital society, they are more likely to be used for carrying secret messages. Hence, more attention is paid on image steganalysis. However, it is a very challenging task because the stego signal introduced to images is rather weak. Currently, a large number of methods have been proposed. Most of these methods follow a two-step paradigm that first computes features from images and then trains a classifier (such as SVM, ensemble classifier [16, 27]) based on these features to distinguish between cover and stego objects. One of the critical issues is how to extract powerful features to capture the traces caused by embedding operations. In this direction, a number of handcrafted features have been proposed in the literature, such as image quality metrics [2], SPAM [36], Rich Models [10] and so on.

Although significant advances have been made in steganalysis in recent years, traditional approaches suffer from some key limitations. Firstly, though the classification module is trainable, the feature extraction module is manually designed, which requires a great deal of human intervention and expertise. To obtain a good feature representation, the steganalyzer needs to carefully consider as many useful statistics as possible. However, it is difficult because typical image statistics are greatly influenced by factors like in-camera processing, image preprocessing and the variation of image contents. In addition, with the rapid development of advanced steganography, the statistical changes caused by embedding operations are much harder to model, which makes the handcrafted feature extraction even more difficult. Secondly, the feature extraction and classification steps are separated in traditional steganalysis systems. Once some useful information is lost in the feature extraction, it can not be recovered in the classification step.

In this paper, we develop a new paradigm for detecting steganography based on the concept of feature learning. In the new methodology, both the feature extraction and classification modules are trainable and are integrated into a single network architecture, which means supervision information from classification step can be utilized to guide feature extraction. It can automatically learn features in a data driven mode. Compared to traditional handcrafted feature based methods, our method can greatly reduce human labor. Our methodology is inspired by the recent development of deep learning, which has become a very hot and trendy machine learning scheme due largely to its powerful capability to automatically learn features from incoming data. Generally, deep learning models are trainable networks consist of multiple levels of non-linear operations. They can be trained using either supervised or unsupervised approaches to learn complex representations by combining information from lower layers. They have been proven to be powerful in learning representations and in understanding image contents for many tasks in practice, especially computer vision tasks [13, 17, 33, 42, 46]. However, unlike the strong visual patterns in these tasks, the stego noise pattern is rather weak and hard to capture. In our work, we leverage domain knowledge of steganalysis, and show how deep learning can be effectively applied to steganalysis.

The main contributions of this paper are highlighted as follows:

1. In this work, we propose a novel CNN based method to automatically learn features for steganalysis. Domain knowledge of steganalysis (i.e. enhancing stego noise and taking advantages of dependencies among nearby pixels) is taken into account when designing CNN architecture to obtain a better performance.
2. Model combination strategy is used to improve the detection performance of CNN based model.
3. We propose an image cropping strategy to enable the proposed model to deal with arbitrary input sizes, especially the large sizes.
4. We visualize the activities within the proposed CNN model to give intuitive understanding of why CNN based model works for steganalysis. The visualization reveals that many intuitively desirable patterns related to steganalysis are automatically captured. We also provide some quantitative analysis of learned features by tapping the learned features and investigating its performance with commonly used ensemble classifier in traditional steganalysis.

This work has evolved from our initial study in [39]. The extension in this work is summarized in the following aspects. Firstly, we investigate the effect of different settings of the deep network structures. Secondly, we provide feature visualization and also some quantitative feature analysis to help understand how the proposed deep network works for steganalysis. Thirdly, we investigate the effect of model combination to boost the detection performance. Fourthly, we consider strategies to enable the proposed deep network to deal with input of arbitrary sizes, especially large sizes. Finally, we conduct experimental evaluation of the proposed deep learning based steganalytic method on more embedding algorithms.

The rest of this paper is organized as follows. In Section 2, we introduce the related work. Section 3 presents the proposed CNN based steganalysis method in detail. In Section 4, we describe the data sources and report our experimental results. Conclusions and future work are given in Section 5.

In the rest of this article, we use the capital-case boldface symbols for matrices and higher-dimensional arrays and the corresponding lower-case letters for their individual elements, i.e., $\mathbf{C} = (c_{ij})^{n_1 \times n_2}$, where n_1 and n_2 are the dimensions. We use $\mathbf{I} \in \{0, ..., 255\}^{n_1 \times n_2}$ to denote an 8-bit gray-scale image.

2 Related work

Traditionally, existing steganalysis methods rely on handcrafted features to model the statistical changes of an image caused by embedding operation. In early period, statistics such as image quality measures (IQMs) [2], moments [14, 19, 34, 54], amplitudes of local extrema in the gray level histogram [6] are used as features. Later, researchers followed a paradigm of modeling the statistical dependencies between adjacent pixels or coefficients by computing Markov process or co-occurrences from the noise residual [7, 36, 44]. The noise residual is obtained by using high-pass filters to strengthen the stego noise, hence making feature representations more sensitive to embedding operations. In recent years, with the increasing sophistication of steganographic methods [1, 20, 21, 31, 37, 43, 55], it is getting harder to to detect accurately when just using simple image models. Hence, researchers proposed to use complex and high order statistics to improve the detection performance [10, 12, 15, 48, 50, 51]. So far, the most representative handcrafted features are the so called rich image

representations [9–11, 15, 22, 45, 48].These features are extracted by firstly using a large number of designed high-pass filters to obtain a family of noise residuals, and then merging features computed from different noise residuals to obtain a high dimensional feature set.

When designing the above handcrafted features, steganalyzers need to specify all of the image statistics that steganalysis systems need, which is time-consuming and laborious. Unfortunately, it is rather difficult to know what features should be extracted for detecting embedding changes because natural images are difficult to model accurately. Moreover, increasing sophistication of embedding algorithms makes the task even more challenging. In our approach, we tackle the problem of feature representation for steganalysis by using deep learning to automatically learn patterns from raw data. We introduc a feature learning framework for steganalysis based on CNNs, one of the representative deep models. In the framework, feature learning and classification are integrated into a single end-to-end CNN network, and are trained as a whole using the backpropagation algorithm. Here, Convolutional Neural Networks are a specialized kind of neural network, in which convolution and pooling operations are applied alternately to the inputs, resulting in increasingly complex feature representations. They have shown excellent performance on feature learning for many visual classification tasks [29]. In our work, the use of CNNs for steganalysis is motivated by one interesting property that the convolutional structure is capable of capturing dependencies among nearby pixels.

In our previous work [39], we propose an CNN based steganalysis framework. Since then, there have been a few CNN based methods [8, 38, 47, 49, 52, 53] for steganalysis. Some of these methods focus on network architecture design. In [38], Pibre et al. present a CNN architecture for steganalysis with fewer convolutional layers, and without the pooling operation. But such architecture is designed specially for steganalysis in the scenario where the embedding changes occur roughly in the same locations over images caused by the fixed embedding key. However, the scenario hardly happens in practice. The method proposed in [8] has the similar problem, which means it works well only when embedding process is operated with same stego key. Xu et al. [53] propose to use absolute activation layer, Tanh activation function and 1×1 convolution in CNN architecture for steganalysis. In [49], the authors propose a method based on Deep Residual Network[17], a kind of very deep CNN model. Compared with these methods, our manuscript have following differences: 1) We provide visualization of feature maps and learned filters to help understand how CNN based model works for steganalysis. 2) We give quantitative feature analysis by tapping learned features from convolutional layers and using them on the ensemble classifier. 3) We propose a strategy to enable the proposed deep network to deal with input of arbitrary sizes, especially large sizes. 4) Model combination is used as a regularization strategy to boost the detection performance. There are some other methods that also use regularization strategies to improve the performance of CNN models. In [40], a transfer learning strategy was applied to improve the performance of CNN on detecting steganography with a low embedding rate. Here, differently, the model combination strategy is effective on improving the detection performance against steganography with high payload as well. In [41], the authors proposed to regularize CNN model with auxiliary features. Different from this method, the proposed model combination strategy does not need extra handcrafted features.

3 Feature learning for steganalysis

In this section, we present the steganalysis method based on CNN for its three advantages. Firstly, CNN is currently one of the most powerful and successful deep models in learning

features automatically. Secondly, the convolutional structure in CNN is suitable for capturing dependencies among pixels, which are important for the steganalysis task. Thirdly, the weight sharing mode in CNN greatly reduces the number of trainable parameters, which enables CNN to deal with large images. However, applying CNN to steganalysis is not straightforward, because CNNs are originally designed to learn patterns related to visual content for computer vision tasks. But in steganalysis, the signal of interest is hidden within the noise component of the image. In the rest of this section, we show how to incorporate the domain knowledge in steganalysis (i.e. enhancing stego noise and exploiting dependencies among nearby pixels) to make CNN work better for steganalysis. The proposed CNN architecture for steganalysis is depicted in Fig. 1.

3.1 CNN based model for steganalysis

The framework of the proposed CNN based steganalysis is shown in Fig. 1. This model takes images as inputs, and is composed of a number of layers including one image processing layer, five convolutional layers for feature representation, and three fully connected layers for classification. Note that the image processing layer is designed explicitly for the steganalysis problem to enhance the stego signal. The whole network is trained using backpropagation to optimize the parameters.

3.1.1 Image processing layer

In the image processing layer, we apply high-pass filtering to preprocess input image. Such preprocessing is typically used in traditional steganalysis [26, 36, 44] to help obtain more compact and robust feature representations. In fact, steganographic methods introduce only small changes to the pixels. Hence, the stego noise pattern is very weak and hard to capture. The use of high-pass filter for preprocessing is to suppress image content and make CNN concentrate on noise components. In this way, the learned feature representations will be more sensitive to stego noise. In our experiments, we found that, without the mandatory high-pass filtering operation, CNN does not converge, which means effective stego noise patterns can not be captured. When using the image processing layer, the training error of CNN quickly drops during training.

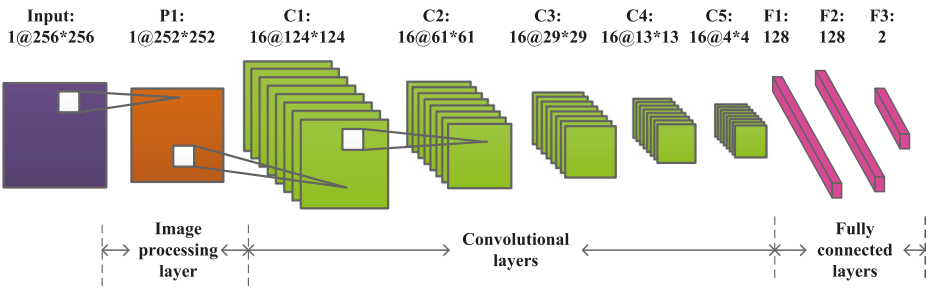

Fig. 1 The proposed CNN based model for steganalysis. It is composed of one image processing layer, five convolutional layers, and three fully connected layers

In our previous work [39], we use the $\mathbf{K}_{5\times5}$ kernel (also called KV kernel) shown as below.

$$\mathbf{K}_{5\times5} = \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix} \tag{1}$$

In this manuscript, we will also investigate some other types of high-pass filters commonly used in traditional steganalysis, including linear and non-linear ones.

3.1.2 Convolutional layer

The feature extraction module consists of five convolutional layers. The filtered image from the image processing layer is fed to the first convolutional layer. At each convolutional layer, as shown in Fig. 2 (right), three kinds of operations i.e. convolution, non-linear activation, and pooling, are usually applied sequentially, generating a number of arrays called feature maps.

The first operation in this layer is convolution, which can be mathematically described as below.

$$\mathbf{X}_j^l = \sum_i \mathbf{X}_i^{l-1} * \mathbf{K}_{ij}^l + \mathbf{B}_j^l, \tag{2}$$

where $\mathbf{X}_j^l$ is the j-th feature map in layer l, $\mathbf{X}_i^{l-1}$ is the i-th feature map in layer $l-1$, $\mathbf{K}_{ij}$ is the trainable filter connecting the j-th output map and the i-th input map, $\mathbf{B}_j^l$ is the bias array for the j-th output map. Note that the input feature map for the first convolutional layer is indeed a filtered image from the image processing layer, while the input feature maps for each of the rest convolutional layers are the output feature maps from its previous layer. Additionally, the weights of filter kernels and the biases have to be learned and updated during training.

Parameter sharing in convolution operation reduces the number of free variables, hence making the network much easier to train [5]. More importantly, the convolution operation is

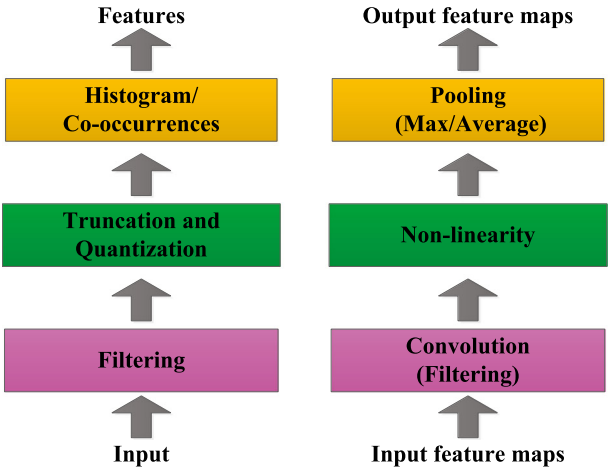

Fig. 2 Components of convolutional layer (right) and traditional steganalysis feature extractor (left)

capable of taking advantage of the key property of images that nearby pixels are dependent. For steganalysis purpose, the property is rather important. In fact, how to capture complex dependencies among neighboring elements of images is a vital consideration in designing modern steganalysis systems [10, 22]. When comparing the components of convolutional layer and traditional steganalysis feature extractor (Fig. 2), we find that both of them use convolution operation. Differently, in the CNN based model, multiple stages of convolution operations are applied. As the network goes deeper, convolution operation in higher convolutional layers successively involves information from larger regions of input images, hence dependencies in larger neighborhood are captured. Similar idea of utilizing dependencies among a large number of dependencies, the so called long range dependencies, for steganalysis can be found in [22]. In that work, two stages of convolution is used to cover a large neighborhood of an input image, hence considering relationships among a large number of pixels. In the first stage convolution, a high-pass kernel is applied to obtain a noise residual. The second stage convolution is called projection, and the elements of projection kernels are randomized from a standard normal distribution. Different from this method, the filter kernels in each convolutional layer are learned rather than picked by hand.

Figure 3 shows some filters learned in the first to fifth convolutional layers in the proposed network on HUGO algorithm with the payload of 0.5 bpp. For the first convolutional layer, we show all of the 16 learned filters of size 5×5. Note that, there are 256 learned filters of size 3×3 in each of the second to fourth convolutional layers, and 256 filters of size 5×5 in the fifth convolutional layer. Here, for each of the second to fifth convolutional layers, we show 16 randomly selected filters due to space constraints. We can observe that the CNN learns high-pass filters, which are more likely to capture stego noise patterns.

After convolution, a non-linear activation function is applied element-wise to the output arrays of convolution operation. In fact, as shown in Fig. 2 (left), traditional steganalysis

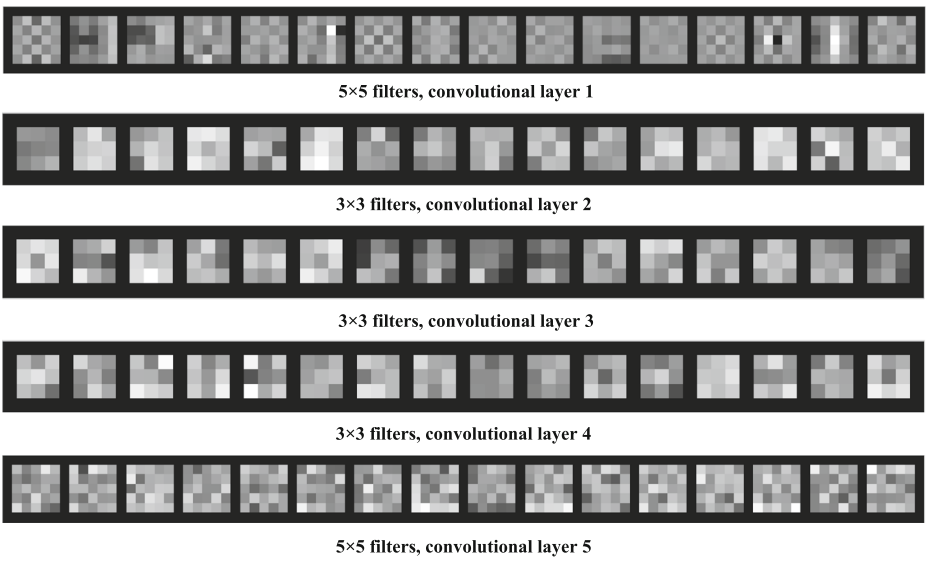

Fig. 3 Examples of filters learned in the first to fifth convolutional layers in the proposed network on HUGO algorithm with the payload of 0.5 bpp. We can observe typical high-pass filters. Especially, some of the learned filters in the first convolutional layer are much similar to the hand-designed filters used in traditional steganalysis methods

extractors also have a non-linearity operation step composed of truncation and quantization. The truncation and quantization are used to restrict residual's dynamic range to reduce the feature dimensionality and make the representations more robust by removing some unpopulated bins. The non-linear activation function here has a similar effect of limiting the amplitude of the output. But more importantly, it gives the network non-linear capabilities, which is important for multi-layer networks to learn complex representations. There are many choices for activation function. In our previous work, we use the Gaussian function below.

$$f(x) = e^{-\frac{x^2}{\sigma^2}}, \tag{3}$$

where σ is a parameter that determines the width of the curve. The Gaussian function is a particular example of radial basis function (RBF), which is commonly used in RBF networks. In this work, many other activation functions are considered, such as the $tanh()$ sigmoid function, the recent Rectified Linear Units (ReLUs) [29], and also a variant of the Gaussian function (referred to as "1-gaussian" in Table 1),

$$f(x) = 1 - e^{-\frac{x^2}{\sigma^2}}. \tag{4}$$

The output arrays of activation form the inputs to the pooling step of the convolutional layer. For each input array of the pooling operation, there is a plane of units in the pooling step and each unit takes inputs from a local region of a size determined by a parameter called pooling size. A pooling function is applied to summarize the responses over the whole corresponding local region. Generally, there are two conventional choices for pooling: average pooling and max pooling. The former takes the average value within the units, while the max pooling operation selects the maximum value. Note that the pooling function is applied to each input array independently. One purpose of pooling is to progressively reduce the spatial size of the feature representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. More importantly, the reduction of spatial resolution of the feature maps means that the features in higher convolutional layers can capture global information from a wider range in the input image. For the task of steganalysis, the stego signal to deal with is extremely weak and hard to represent with statistics computed from a small local region. It is important to exploit more global statistics from a large region of the input image to obtain more efficient representations. In fact, in traditional steganalysis systems as shown in Fig. 2 (left), histograms and co-occurrences, are usually computed to capture global information. But in this paper, global information in the proposed CNN based model is captured with multiple layers of pooling operation to progressively involve information from increasingly large regions of the input image. In our experiments, we test the CNN network structures with and without pooling operation, respectively, in the scenario of detecting the representative S-UNIWARD steganographic algorithm with unfixed random embedding keys. We indeed experimentally observe that the CNN network structure with pooling operation outperforms the one without pooling operation as proposed in [38].

Besides the three operations discussed above, normalization technique is often applied after pooling, and many types of normalization methods, such as local response normalization [29] and local contrast normalization [24], have been proposed in CNN architectures.

Table 1 Detection error of CNN model against HUGO algorithm when using different activation functions and pooling operations

Pooling	Gaussian	1-Gaussian	ReLU	Tanh
Average	17.20%	16.55%	16.65%	17.28%
Max	19.05%	19.65%	19.68%	18.95%

Generally, normalization in CNN can be regarded as a kind of regularization technique to introduce competitions to neuron outputs. It can reduce the correlations among the data, and make the features in feature maps more significant, hence improving the training of CNN. In our experiments, we investigate the effect of local response normalization and local contrast normalization, respectively. But the result shows that both kinds of normalization have little impact. In fact, both normalization methods have also fallen out of favor recently in many other tasks. It may be because their role has been outplayed by other regularization techniques such as dropout [18] and better initializations.

To summarize, in a CNN model with multiple convolutional layers, each convolutional layer extracts features from the previous layer. As the network goes deeper, complex dependencies are captured by progressively involving larger regions of the input image. Hence, the information is gathered hierarchically from local to global to make the neurons in the top layer predict with higher confidence whether the image is modified or not.

3.1.3 Classification layer

The classification module consists of several fully connected layers. The extracted features from convolutional layers are passed to this module. On the top layer, a softmax activation function is used to produce a distribution over all the class labels. In our network, a two-way softmax is used as below.

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{2} e^{x_j}}, \tag{5}$$

for $i = 1, 2$, where x_i is the total input to the neuron i in the top layer, and y_i is its output.

To reduce the problem of overfitting, a recently proposed technique called "dropout" is applied for regularizing fully connected layers [18]. When training with dropout, the output of each neuron in corresponding layers is set to zero with probability 0.5. This technique improves the network's generalization ability and test performance.

The CNN is trained by minimizing $-logy_t$, where $t \in \{1, 2\}$ denotes the target class, using back-propagation algorithm. Error propagation and weight adaptation in convolutional and fully connected layers follow the standard procedure. Details of the procedure are given in Ref. [30]. In a word, all the parameters in feature extraction and classification modules are jointly optimized.

3.2 Model combination

Despite the discriminative power of large Convolutional Neural Networks, they are prone to overfit, especially when the amount of training data is limited. In practice, a wide range of techniques for regularization have been developed to tackle this issue, and model combination is usually an important regularization strategy for combating overfitting and reducing CNN based generalization error. In this paper, we propose a simple model combination method for CNN based steganalysis.

Firstly, we train several different CNN models separately using the same network architecture but with different types of high-pass kernels used independently in the image processing layer. Note that, in the previous work [39], only a non-directional linear kernel $\mathbf{K}_{5\times5}$ is used. In this work, besides the $\mathbf{K}_{5\times5}$ kernel, three other kernels $\mathbf{K}_{3\times2}$, $\mathbf{K}_{1\times4}$ and $\mathbf{K}_{5\times5}^{max}$ are also used. $\mathbf{K}_{3\times2}$ and $\mathbf{K}_{1\times4}$ are directional filters that help capture more information about edges and complex textures. $\mathbf{K}_{5\times5}^{max}$ is a non-linear filter which takes the the maximum of four $\mathrm{K}_{3\times5}$ or $\mathrm{K}_{5\times3}$ linear filters' outputs. It introduces non-linearity into the residuals. More details of these kernels can be found in [10]. When training different CNN models

separately using different high pass kernels, it is supposed that different embedding artifacts can be captured. In fact, in [10], a large family of high-pass filters are exploited during handcrafted feature extraction. The reason for constraining our investigation to only four of these kernels is that the focus of this paper is on the methodology rather than benchmarking steganographic methods on some data sources. In fact, our goal here is to investigate the effectiveness of incorporating prior knowledge on features for CNN based steganalysis via model combination.

Secondly, after training these different CNN models, we combine these models to encode the complementary information. During combination, we first select the model with the lowest detection error. Then one by one, we add the model among the remaining models that leads to the biggest drop of the detection error for combination until the detection error no longer drops. The strategy continuously takes advantage of classification feedback of model combination to greedily minimize the detection error in every iteration. Note that, 'combination' means that output probabilities (soft outputs) of the combined models are averaged, and the result is compared with the threshold of 0.5 to determine the corresponding class label.

4 Experiments

4.1 Dataset

In this paper, experiments are carried out on the standard database called BOSSbase 1.01 [3] to evaluate the effectiveness of the proposed CNN based framework. This database contains 10,000 images acquired by seven digital cameras in Raw format and subsequently processed to the size of 512×512 pixels.

In our work, we randomly split the dataset by assigning 80% of the images (covers) to a training set and 20% to a testing set, and keep the split fixed for all later experiments. After embedding with a given steganographic algorithm and a specified embedding rate, images of the other class (stego) can be generated. Hence, the training set and the testing set obtained from BOSSbase 1.01 consists of 8000 and 2000 cover/stego pairs respectively. To evaluate the performance of our designed CNN models, state-of-the-art steganographic algorithms, such as HUGO [37], WOW [20], S-UNIWARD [21], MiPOD [43], and HILL-CMD [31] are considered. These steganographic algorithms are implemented with unfixed random embedding keys in our experiments. We compare the performance of the proposed CNN model with state-of-the-art handcrafted feature sets, such as SRM [10] and maxSRMd2 [9], implemented with ensemble classifier. Note that the experiments using handcrafted feature sets (with ensemble classifiers) are conducted on the same training/testing split as for the CNNs.

4.2 Data preprocessing

In many application areas of deep learning, data preprocessing plays an important role as it can transform the data into a form that will be more easily and effectively processed for later steps such as feature representation and classification. For example, data preprocessing skills, such as per-example mean subtraction [25] and whitening [35], are used in many computer visual tasks to help deep learn algorithms obtain better features.

In our work, besides the high pass filtering we mentioned in the former section, another preprocessing strategy is also used for the steganalysis task. We crop the input images into patches of the same size, and feed the patches to CNN based model, hence enabling the

CNN based detector to deal with arbitrary input sizes, especially large sizes. The motivation behind the second data preprocessing strategy is that a CNN architecture requires fixed input image size due to the existence of fully-connected layers. Additionally, the larger the input size, the greater memory GPU requires, and the more parameters the CNN based model need to train. In practice, it is common that the steganalyzer needs to detect images with arbitrary sizes. Especially, some of the images are with a large size (e.g. 512×512, 2000×3000), hence it is important that CNN based approach can deal with images of various sizes, especially large sizes. Generally, in computer vision tasks, this problem can be solved by simply resizing both the training and testing images to a standard size (e.g. 256×256) while retaining the important properties of data corresponding to the tasks. However, in steganalysis, resizing operation will certainly erase the stego signal. Hence, instead, we crop the input images into patches to reduce and fix the size of input while retaining traces caused by steganographic modification.

In the training phase, we firstly crop a raw input image (cover or stego) from the training set into a number of image patches with a fixed size. We then feed these patches to the network for training. In the testing phase, we firstly extract image patches from each cover or stego in the testing set with the same method used in the training phase, and then propagate each patch individually to the trained network. Finally, we opt for the simplest approach of averaging the output probabilities output by the networks softmax layer on each of these patches to produce the final estimate of the class probabilities for the entire image. This idea can be interpreted as an ensemble approach to utilize information from different parts of the raw image. We expect more complicated techniques to further improve performance but consider these to be out of the scope of this paper. Actually, this way of learning and testing has been commonly used in CNN based models for many other tasks [25, 29], in which it is used as a data augmentation strategy to reduce overfitting problem. Here differently, we borrow this idea mainly for the purpose of dealing with large input size.

In our experiments on BOSSbase 1.01, due to the GPU memory limitation, it is hard for our proposed deep network to directly use the images (cover or stego) of size 512×512 as inputs. Hence, we use the cropping strategy described above on this dataset. In the training phase, we extract five 256×256 patches (center patch and four corner patches) and their horizontal reflections (hence ten patches in total) from each image (cover or stego) of size 512×512 in the training set, and present the ten image patches to the network for training. Here, a horizontal reflection is obtained by flipping the image patch horizontally. Note that the use of horizontal reflections is to further enlarge the amount of data to reduce overfitting. After this step, 80,000 cover/stego patch pairs of size 256×256 are generated from the 8,000 cover/stego pairs of size 512×512, hence increasing the size of the training set by a factor of 10. Training on image patches of a smaller size greatly reduces the GPU memory requirement. In the testing phase, we generate ten patches from each cover or stego in the testing set using the same strategy as in the training phrase, and average the ten output probabilities on each 256×256 patches to make the final prediction.

The complexity of the convolutional feature computation when training on image patches of size 256×256 is $O(n \cdot 256^2)$, where n is the number of patches per image. In our experiments, n is 10. Note that, when training on whole images of size 512×512, this complexity is $O(512^2)$.

4.3 Network architecture settings

As shown in in Fig. 1, the proposed network is composed of one image processing layer, five convolutional layers and three fully connected layers. It accepts an input image (or

image patch) of size 256×256. The image processing layer filters the input image with a predefined high-pass filter to obtain a residual. Note that, we do not use padding here. The size of output images from the image processing layer is reduced to 252×252. Each of the five convolutional layers generates 16 feature maps. The first convolutional layer takes the residual from the image sprocessing layer as input and filters it with 16 trainable kernels of size 5×5. The second, third and fourth convolutional layers apply convolutions with the kernel size of 3×3. The size of convolution kernel used in the fifth convolutional layer is 5×5. The filtering stride of all convolution operations in the five convolutional layers is 1. At each convolutional layer, the non-linearity function is applied element-wise to the output of convolution operation. Moreover, each of the five convolutional layers applies an overlapping pooling operation with the window size 3×3 and stride 2. The choices of high-pass kernels in the processing layer, and the activation function, pooling operation in the convolutional layers will be discussed in the later parts of the experimental section.

After five layers of convolution and pooling operations, the input image has been converted into a 256D feature vector capturing the steganographic traces. Finally, the extracted 256 features are passed to the classification module consisting of three fully connected layers. Each of the first two fully connected layers has 128 neurons. For classification, the output of each neuron in the first two fully connected layers is activated by ReLUs. The last fully connected layer has the same number of neurons as the number of classes to classify. Since steganalysis is a binary classification problem, the number of neurons is 2 in the last fully connected layer, and the outputs of the last fully connected layer are fed to a two-way softmax.

4.4 Training

The training of our proposed network is carried out using the code provided by Krizhevsky et al. [28], which allow for rapid experimentation. In our experiments, a Tesla K40c GPU with 12GB of memory is used.

The number of trainable parameters in the five convolution layers is 13792 in total. These parameters come from the convolution weights and the biases. When including the fully connected layers, the total number of the trainable parameters for the whole network is 63456. All the weights in each convolutional and each fully connected layer are initialized from a zero-mean Gaussian distribution with standard deviation of 0.1. The neuron biases in each layer are initialized with the constant 0. We train our CNN models using mini-batch stochastic gradient descent with a mini-batch size of 128 examples. It means a mini-batch of 128 images (or image patches) of size 256 from the training set is input for each iteration. Note that the covers and stegos are not necessary paired in the mini-patch. The weight decay is 0 for the convolutional layers and 0.01 for the fully connected layers. The momentum is fixed to 0.9. When using the Gaussian or '1-Gaussian' activation function, we fixe the parameter σ to 1 in the first convolutional layer, and 0.5 in the second to fifth convolutional layers. All models are initialized with learning rates of 0.001. For the BOSSbase 1.01 dataset, we first train on 80% cover/stego pairs of the training set, and use the rest of the training set for validation. After validation error no longer decreases, we then train on all the training set until the validation error is near the training error. We further lower learning rates by a factor of 10 for two times, and train about 10 epoches each time. The total number of epoches needed vary from 200 to 300.

4.5 Steganalysis on BOSSbase 1.01

In this section, we first evaluate the steganalysis performance of the CNN based model under different settings of activation functions and pooling methods on BOSSbase 1.01. Then we investigate the effectiveness of model combination and feature combination to fuse information from different residuals.

4.5.1 Effect of varying activation functions and pooling operations

In this experiment, we analyze the effects of using four different activation functions (Gaussian, 1-Gaussian, ReLU, Tanh) and two pooling operations (average, max) in the proposed network as described above. The kernel for preprocessing in the image processing layer is fixed to be the $\mathbf{K}_{5\times5}$ kernel, one of the commonly used kernels in traditional steganalysis. Table 1 shows the detection error of CNN models against HUGO algorithm with a fixed payload of 0.4 bpp (bits per pixel). We can notice that average pooling outperforms max pooling. Generally, max pooling is well suited to feature representations that are very sparse (i.e. have a very low probability of being active) [4]. In average pooling, all the activations in the pooling region are taken into account, which is supposed to discard the disturbances caused by individual elements. We also find that the considered four kinds of activation functions have rather similar performance. In the rest of the paper, we use average pooling and the '1-Gaussian' activation function in the convolutional layers of the CNN models.

4.5.2 Effect of varying the depth of CNN architecture

In this section, we evaluate the effects of varying the number of convolutional layers and fully connected layers. We do this by withdrawing different number of convolutional layers or fully connected layers from the designed network architecture (Fig. 1), while keeping the rest of network settings untouched. Table 2 shows results on detecting HUGO and SUNIWARD algorithms with payload of 0.4 bpp on BOSSbase 1.01. An increased detection error can be seen as we withdraw more convolutional layers. In addition, the increase of detection error caused by withdrawing the first two fully connected layers is not significant. The best result is obtained by retaining all layers. These results support the argument that as the network architecture goes deeper, increasingly more powerful features can be learned.

4.5.3 Feature visualization

In order to provide intuitive interpretation of how the proposed model works, we visualize the activity within the model. We do this by plotting the feature maps produced in each layer of a trained CNN for a given cover or stego image. Although this kind of visualization method is simple to implement, we do find it informative because all data passing through the network can be clearly visualized.

Figure 4 shows the features extracted from the trained CNN model against HUGO with payload of 0.5 bpp. The features shown in the first and the second row are associated with a cover patch (row 1, column 1) and the corresponding stego patch (row 2, column 1) from the testing set, respectively. The cover patch and stego patch are preprocessed using KV kernel in the image processing layer to obtain noise residuals (row 1 & 2, column 2). In the third to seventh columns, we visualize the feature maps obtained from each convolutional layer in

Table 2 Detection error of CNN against HUGO and SUNIWARD when withdrawing different layers form the proposed CNN architecture

	Withdraw C2-C5	Withdraw C3-C5	Withdraw C4-C5	Withdraw C5	Withdraw F1-F2	Proposed
HUGO 0.4bpp	23.65%	19.80%	18.25%	16.95%	16.72%	16.55%
SUNIWARD 0.4bpp	30.40%	26.83%	24.80%	24.43%	24.33%	24.20%

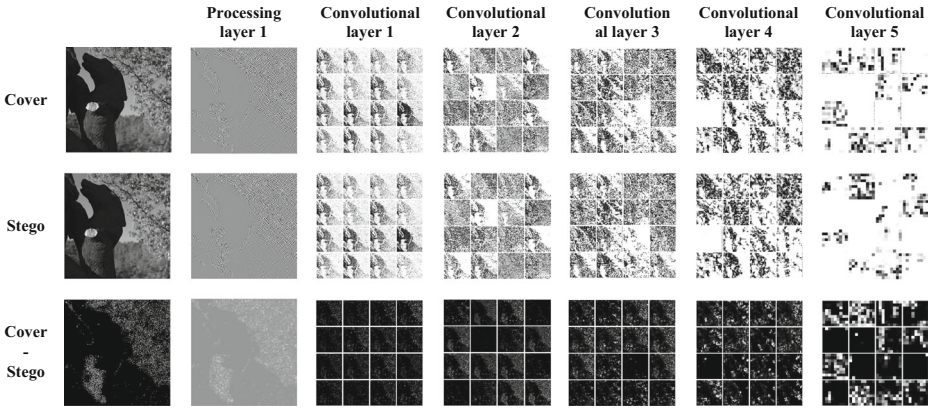

Fig. 4 Visualization of feature maps extracted from the trained CNN model against HUGO algorithm with the payload of 0.5 bpp. The features shown in the first and the second rows are associated with a cover patch (row 1, column 1) and the corresponding stego patch (row 2, column 1) from the testing set, respectively. The cover patch and stego patch are preprocessed using KV kernel in the image processing layer to obtain noise residuals (row 1 & 2, column 2). In the third to seventh columns, we visualize the feature maps obtained from each convolutional layer in the first and the second row, respectively. In addition, the differences between the images or the feature maps in the first row and the second row are shown in the third row of the corresponding columns

the first and the second row, respectively. Note that there are 16 feature maps obtained from each convolutional layer. In addition, the absolute difference between the images or the feature maps in the first and the second row is shown in the third row of the corresponding column.

We observe that the learned features are neither random, uninterpretable patterns nor visual patterns. Rather, they show many desirable properties related to steganographic embedding. Firstly, the feature maps of each convolutional layer correspond to noise areas where embedding modification are mainly made. Secondly, as shown clearly in the last row, the difference between cover feature maps and stego feature maps in top layers is more significant than in bottom layers, which indicates that features learned in top convolutional layers is more discriminative.

4.5.4 Effect of model combination

To evaluate the effectiveness of model combination, for each steganographic algorithm with a fixed payload, we train four CNN based models $K_{5\times5}$-CNN, $K_{3\times2}$-CNN, $K_{5\times5}^{max}$-CNN, and $K_{1\times4}$-CNN with four types of kernels used independently in the image processing layer, and then combine them to the final prediction. We report the detection error of single CNN model and model combination on detecting four state-of-the-art spatial domain steganographic algorithms: WOW, S-UNIWARD, MiPOD [43], and HILL-CMD [31], with payloads 0.3, 0.4, and 0.5 bpp, respectively, in Table 3. We can observe that the detection error of these individual models is slightly different, and model combination boosts the detection performance. Especially, the model combination based method brings 1-3% drop of detection error when compared to $K_{5\times5}$-CNN proposed in our previous work. These results indicate the effectiveness of our model combination scheme.

When compared with SRM implemented with ensemble classifiers (SRM + EC), the detection error of our method is comparable on detecting WOW, but is 1–4% higher on

Table 3 Detection error of single CNN models and model combination against four content-adaptive steganographic algorithms

Algorithm	WOW			S-UNIWARD			MiPOD			HILL-CMD		
Payload (bpp)	0.3	0.4	0.5	0.3	0.4	0.5	0.3	0.4	0.5	0.3	0.4	0.5
$K_{5\times5}$-CNN	28.93%	21.98%	17.35%	32.05%	24.20%	20.65%	32.78%	27.63%	22.83%	38.55%	33.35%	29.75%
$K_{3\times2}$-CNN	27.45%	21.80%	16.75%	31.38%	22.30%	19.03%	32.98%	26.38%	20.93%	36.95%	33.25%	29.13%
$K^{max}_{5\times5}$-CNN	29.73%	22.58%	19.08%	31.78%	23.48%	19.70%	33.13%	27.68%	23.55%	37.68%	34.30%	30.70%
$K_{1\times4}$-CNN	27.70%	20.55%	16.68%	30.60%	22.35%	17.68%	32.28%	27.13%	21.78%	36.88%	31.88%	29.00%
CNN model combination	26.97%	20.05%	16.03%	30.38%	21.72%	17.45%	32.27%	26.07%	20.92%	36.88%	31.70%	28.30%
$K_{5\times5}$-SRM + EC	31.95%	26.72%	22.45%	30.60%	26.25%	22.25%	33.22%	28.53%	24.10%	39.00%	35.13%	31.45%
SRM + EC	25.57%	20.90%	16.60%	26.12%	20.92%	16.70%	29.05%	24.32%	20.42%	34.65%	29.83%	26.07%
$K_{5\times5}$-maxSRMd2 + EC	25.67%	21.82%	18.93%	30.48%	26.35%	22.13%	32.85%	29.03%	24.87%	35.35%	31.03%	28.87%
maxSRMd2 + EC	18.77%	15.53%	12.84%	23.62%	19.51%	15.66%	27.52%	22.43%	19.00%	30.28%	26.32%	23.40%
Xu et al.	27.04%	22.08%	19.23%	25.72%	20.70%	18.15%	29.12%	23.72%	20.07%	35.28%	30.42%	27.10%

The results are compared with representative handcrafted feature sets implemented with ensemble classifier (EC)

detecting S-UNIWARD, MiPOD, and HILL-CMD. One possible reason for the lower accuracy of our method is that our method uses only four kinds of high pass kernels for preprocessing, while more than 100 kinds of kernels are used in SRM feature extraction. These filters are used to obtain different noise residuals. Then by merging features computed from a large number of residuals, the possibility of capturing embedding artifacts is greatly increased. In our method, due to time and hardware limitations, we only consider four kinds of high pass kernels for model combination. However, we find that, when both our CNN based method and handcrafted feature extraction use the same high pass kernel for preprocessing, our method performs better. In Table 3, $K_{5\times5}$-SRM and $K_{5\times5}$-maxSRMd2 are subsets of SRM and maxSRMd2, respectively. Both $K_{5\times5}$-SRM and $K_{5\times5}$-maxSRMd2 are extracted from noise residual computed using $K_{5\times5}$ kernel. The dimensionality of them is 507. We implement these feature sets individually with ensemble classifier (EC), and then compare the detection error of them with $K_{5\times5}$-CNN. It can be observed that our $K_{5\times5}$-CNN outperforms $K_{5\times5}$-SRM + EC and achieves comparable performance with $K_{5\times5}$-maxSRMd2 + EC, which shows the effectiveness of feature learning using CNN. The comparison of detection error between $K_{5\times5}$-SRM + EC and SRM + EC indicates that some useful information are lost when using only one high pass kernel for preprocessing. Another reason is that the ensemble classifier used in traditional steganalysis is more powerful than the softmax classifier in a CNN model. Actually, experimental results in next subsection bear out this supposition. Additionally, when compared with the maxSRMd2 implemented with ensemble classifiers (maxSRMd2 + EC), the detection error of our method is 2-6% higher on detecting S-UNIWARD, MiPOD, and HILL-CMD, and 3-8% higher on detecting WOW. The reason for much lower detection error of maxSRMd2 is that selection channel (embedding probability) information is exploited during feature extraction. In our future work, we will also try to utilize this priori knowledge to our CNN based method to obtain better performance.

Moreover, we compare our method with Xu et al's method [53], which is one of the representative CNN based steganalysis methods. In that method, $K_{5\times5}$ is also used for preprocessing. The proposed $K_{5\times5}$-CNN [39] performs better on detecting WOW, but worse on detecting S-UNIWARD, MiPOD and HILL-CMD. Compared with the $K_{5\times5}$-CNN architecture, Xu et al's CNN architecture uses more advanced CNN structures proposed in recent years, such as batch normalization [23], 1×1 convolution [32] and global average pooling [32]. Different from Xu et al's work that focuses on structural design, this manuscript propose a model combination strategy. We notice that when applying model combination strategy to our previous method, the detection performance becomes more close to Xu et al's method. And we believe that the model combination strategy can also be applied to other CNN based steganalysis methods, like Xu et al's model, to further improve the performance.

4.5.5 Feature analysis

In this section, we further give some quantitative analysis about the learned features. We extract the features learned from different convolutional layers in each CNN based model and use them with ensemble classifier which is a powerful classifier commonly used in steganalysis. To represent a whole image of size 512×512, we first generate 10 image patches of size 256×256 in the same manner we have described before, and then combine features extracted from each of the ten patches. Note that, since center patch of an image (or its horizontal reflection) is partly overlapped with each of the four corner patches, there is a redundancy in the obtained features. But in our experiments, we find that the impact of this kind of redundancy is minor.

Table 4 Detection error of learned features from convolutional layer C3, C4 and C5 implemented with ensemble classifiers

Algorithm	WOW			S-UNIWARD			MiPOD			HILL-CMD		
Payload (bpp)	0.3	0.4	0.5	0.3	0.4	0.5	0.3	0.4	0.5	0.3	0.4	0.5
C3 fea. + EC	30.25%	24.18%	21.00%	31.82%	26.92%	19.60%	33.87%	29.10%	23.33%	38.53%	33.15%	28.52%
C4 fea. + EC	28.05%	21.05%	18.57%	29.63%	23.57%	18.65%	32.35%	26.62%	21.07%	36.58%	30.90%	26.62%
C5 fea. + EC	28.07%	22.03%	17.27%	30.28%	24.15%	18.94%	31.70%	25.22%	21.22%	37.48%	31.80%	27.82%

Table 5 Detection error of learned features and handcrafted features implemented with ensemble classifiers

Algorithm	WOW			S-UNIWARD			MiPOD			HILL-CMD		
Payload (bpp)	0.3	0.4	0.5	0.3	0.4	0.5	0.3	0.4	0.5	0.3	0.4	0.5
$K_{5\times5}$-CNN fea. + EC	28.07%	22.03%	17.27%	30.28%	24.15%	18.94%	31.70%	25.02%	21.22%	37.48%	31.80%	27.82%
$K_{3\times2}$-CNN fea. + EC	27.17%	20.57%	16.20%	30.35%	22.58%	18.55%	31.95%	24.82%	20.08%	38.01%	31.90%	36.83%
$K^{max}_{5\times5}$-CNN fea. + EC	28.78%	22.45%	17.70%	30.65%	23.92%	18.07%	32.70%	26.02%	21.80%	37.57%	33.02%	26.38%
$K_{1\times4}$-CNN fea. + EC	27.32%	19.68%	16.30%	29.88%	23.18%	20.92%	31.20%	26.27%	20.80%	36.15%	30.90%	26.10%
CNN-4 fea. + EC	26.12%	18.70%	14.82%	28.70%	21.65%	17.32%	30.75%	24.67%	19.25%	35.85%	29.55%	25.37%
$K_{5\times5}$-SRM + EC	31.95%	26.72%	22.45%	30.60%	26.25%	22.25%	33.22%	28.53%	24.10%	39.00%	35.13%	31.45%
SRM + EC	25.57%	20.90%	16.60%	26.12%	20.92%	16.70%	29.05%	24.32%	20.42%	34.65%	29.83%	26.07%
$K_{5\times5}$-maxSRMd2 + EC	25.67%	21.82%	18.93%	30.48%	26.35%	22.13%	32.85%	29.03%	24.87%	35.35%	31.03%	28.87%
maxSRMd2 + EC	18.77%	15.53%	12.84%	23.62%	19.51%	15.66%	27.52%	22.43%	19.00%	30.28%	26.32%	23.40%
CNN-SRM fea. + EC	23.65%	17.15%	13.35%	24.71%	18.80%	14.40%	24.17%	17.77%	12.85%	32.00%	26.77%	23.30%
CNN-maxSRMd2 fea. + EC	19.69%	15.07%	11.88%	23.80%	17.78%	13.96%	22.45%	17.15%	12.37%	28.83%	24.55%	21.32%

In the first set of experiments, we investigate the performance of learned features from different convolutional layers in a single CNN model by implementing them with ensemble classifier. Here, the features are all extracted from $K_{5\times5}$-CNN model. According to the architectural settings, with input image patch of size 256×256, we can obtain 13456, 2704 and 256 features, respectively, from the third, fourth and fifth convolutional layers C3, C4 and C5. For an image of size 512×512 in BOSSbase 1.01, the features obtained from the third, fourth and fifth convolutional layers are with the dimensionality of 134560, 27040 and 2560, respectively. We report the detection error of these feature sets implemented with ensemble classifiers in Table 4. It can be observed that, learned features from C4 and C5 layers provide a lower detection error over features from C3 layer, which indicates features learned from high layer is more effective. The detection error of features from C5 is a little higher than features from C4, but the dimensionality is much lower.

In the second set of experiments, we investigate the performance of learned features from different CNN models and feature combination using ensemble classifier. We report the detection error in Table 5 against WOW, S-UNIWARD, MiPOD and HILL-CMD with payloads 0.3, 0.4, and 0.5 bpp, respectively. Note that the learned features used here is obtained from C5 layer of $K_{5\times5}$-CNN, $K_{3\times2}$-CNN, $K^{max}_{5\times5}$-CNN, and $K_{1\times4}$-CNN, respectively. Compared with the results shown in Table 3, we can observe that the detection error for ensemble classifier is lower than softmax classifier used in CNN models. We can also observe that the learned features from $K_{5\times5}$-CNN outperforms $K_{5\times5}$-SRM and achieves comparable performance with $K_{5\times5}$-maxSRMd2. Note that, $K_{5\times5}$-SRM and $K_{5\times5}$-maxSRMd2 are extracted from noise residual computed using $K_{5\times5}$ kernel. Furthermore, by combing learned features obtained from four CNN models together, we obtain the CNN-4 feature set with the dimensionality of 10240. The results of CNN-4 feature set show the combination of learned features boosts detection performance. It indicates that the CNN models trained with different high-pass kernels used in the image processing layer are able to encode complementary information. Finally, by combining the CNN-4 feature set with the handcrafted SRM and maxSRMd2, respectively, we can obtain the CNN-SRM and the CNN-maxSRMd2 feature sets with the dimensionality of 44911. According to the results in Table 5, combination of learned features and handcrafted features further improve the detection performance, which indicates the learned features do capture some useful information about steganalysis that is different from handcrafted features. We emphasize that each element of the features learned from CNN based models for steganalysis is the response of an image region while the handcrafted features for steganalysis are global statistics computed over the entire image.

5 Conclusion

In this paper, we present a novel feature learning based method for steganalysis. The proposed method is based on Convolutional Neural Network, one of the most representative deep learning models. Different from traditional steganalysis schemes, feature extraction and classification steps are unified under a single network architecture to jointly optimize all the parameters in both steps. It means that the supervision information from classification can be utilized to guide the feature extraction step. Moreover, we propose to exploit regularization strategies, such as data preprocessing and ensemble of networks, to make CNN work better for the steganalysis task. Feature visualization reveals that the learned representations capture some noise-like patterns corresponding to stego signal. Experimental results on BOSSbase 1.01 verify that a well-designed CNN based model can be effectively applied

for detection of weak patterns introduced by steganographic embedding, which indicates automatizing steganalysis via deep models is feasible and promising. We believe that the proposed CNN based method can find applications beyond image steganalysis in related fields, such as audio and video steganalysis, and digital forensics.

In the future, we will focus on improving the performance of CNN based method for steganalysis. Though the performance of current CNN based methods still could not outperform advanced handcrafted feature based methods, we believe that the design of some modules in the proposed method certainly deserves further optimization that might further improve the performance. For example, current method needs a hard wired image processing layer for preprocessing, which means the features are learned on noise residuals rather than the original images. Replacing this fixed module with a learnable one may improve the detection performance. Additional boost can likely be obtained by incorporating other prior knowledge of steganalysis like selection channel information into CNN based models.

Acknowledgments This work was supported in part by the National Natural Science Foundation of China under Grant 61772529, 61303262, 61502496, U1536120, and U1636201, and in part by the National Key Research and Development Program of China under Grant 2016YFB1001003.

References

1. Atawneh S, Almomani A, Al Bazar H, Sumari P, Gupta B (2016) Secure and imperceptible digital image steganographic algorithm based on diamond encoding in dwt domain. Multimed Tools Appl 76(18):18451–18472
2. Avcibas I, Memon N, Sankur B (2003) Steganalysis using image quality metrics. IEEE Trans Image Process 12(2):221–229
3. Bas P, Filler T, Pevnỳ T (2011) Break our steganographic system: the ins and outs of organizing boss. In: Information hiding. Springer, Berlin, pp 59–70
4. Boureau YL, Ponce J, LeCun Y (2010) A theoretical analysis of feature pooling in visual recognition. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 111–118
5. Browne M, Ghidary SS (2003) Convolutional neural networks for image processing: an application in robot vision. In: AI 2003: advances in artificial intelligence. Springer, Berlin, pp 641–652
6. Cancelli G, Doërr G., Cox IJ, Barni M (2008) Detection of±1 lsb steganography based on the amplitude of histogram local extrema. In: 15Th IEEE international conference on image processing. IEEE, Piscataway, pp 1288–1291
7. Chen C, Shi YQ (2008) Jpeg image steganalysis utilizing both intrablock and interblock correlations. In: IEEE International symposium on circuits and systems. IEEE, Piscataway, pp 3029–3032
8. Couchot JF, Couturier R, Guyeux C, Salomon M (2016) Steganalysis via a convolutional neural network using large convolution filters for embedding process with same stego key. arXiv:1605.07946
9. Denemark T, Sedighi V, Holub V, Cogranne R, Fridrich J (2014) Selection-channel-aware rich model for steganalysis of digital images. In: 2014 IEEE international workshop on information forensics and security (WIFS). IEEE, Piscataway, pp 48–53
10. Fridrich J, Kodovsky J (2012) Rich models for steganalysis of digital images. IEEE Trans Inf Forensics Secur 7(3):868–882
11. Fridrich J, Kodovskỳ J, Holub V, Goljan M (2011) Steganalysis of content-adaptive steganography in spatial domain. In: Information hiding. Springer, Berlin, pp 102–117
12. Geetha S, Sindhu SSS, Kamaraj N (2009) Blind image steganalysis based on content independent statistical measures maximizing the specificity and sensitivity of the system. Comput Secur 28(7):683–697
13. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
14. Goljan M, Fridrich J, Holotyak T (2006) New blind steganalysis and its implications. In: Electronic imaging 2006. International society for optics and photonics, Bellingham, pp 607,201–607,201
15. Gul G, Kurugollu F (2011) A new methodology in steganalysis: breaking highly undetectable steganograpy (hugo). In: Information hiding. Springer, Berlin, pp 71–84

16. He FY, Chen TS, Zhong SP (2015) A classifier ensemble algorithm based on improved rsm for high dimensional steganalysis. Journal of Information Hiding and Multimedia Signal Processing 6(2):198–210
17. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
18. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580
19. Holotyak T, Fridrich J, Voloshynovskiy S (2005) Blind statistical steganalysis of additive steganography using wavelet higher order statistics. In: Communications and multimedia security, vol. 3677, pp. 273–274
20. Holub V, Fridrich J (2012) Designing steganographic distortion using directional filters. In: The IEEE international workshop on information forensics and security (WIFS), pp 234–239
21. Holub V, Fridrich J (2013) Digital image steganography using universal distortion. In: Proceedings of the first ACM workshop on information hiding and multimedia security. ACM, New York, pp 59–68
22. Holub V, Fridrich J (2013) Random projections of residuals for digital image steganalysis. IEEE Trans Inf Forensics Secur 8(12):1996–2006
23. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167
24. Jarrett K, Kavukcuoglu K, Lecun Y et al. (2009) What is the best multi-stage architecture for object recognition? In: 2009 IEEE 12Th international conference on computer vision. IEEE, Piscataway, pp 2146–2153
25. Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L (2014) Large-scale video classification with convolutional neural networks. In: IEEE Conference on computer vision and pattern recognition, pp 1725–1732
26. Ker AD, Böhme R. (2008) Revisiting weighted stego-image steganalysis. In: Electronic imaging 2008. International society for optics and photonics, Bellingham, pp 681,905–681,905
27. Kodovsky J, Fridrich J, Holub V (2012) Ensemble classifiers for steganalysis of digital media. IEEE Trans Inf Forensics Secur 7:432–444
28. Krizhevsky A (2012) cuda-convnet. Http://code.google.com/p/cuda-convnet/
29. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
30. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
31. Li B, Wang M, Li X, Tan S, Huang J (2015) A strategy of clustering modification directions in spatial image steganography. IEEE Trans Inf Forensics Secur 10(9):1905–1917
32. Lin M, Chen Q, Yan S (2013) Network in network. arXiv:1312.4400
33. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: Single shot multibox detector. In: European conference on computer vision. Springer, Berlin, pp 21–37
34. Lyu S, Farid H (2003) Detecting hidden messages using higher-order statistics and support vector machines. In: Information hiding. Springer, Berlin, pp 340–354
35. Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY (2011) Multimodal deep learning. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 689–696
36. Pevny T, Bas P, Fridrich J (2010) Steganalysis by subtractive pixel adjacency matrix. IEEE Trans Inf Forensics Secur 5(2):215–224
37. Pevnỳ T, Filler T, Bas P (2010) Using high-dimensional image models to perform highly undetectable steganography. In: Information hiding. Springer, Berlin, pp 161–177
38. Pibre L, Pasquet J, Ienco D, Chaumont M (2016) Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source-mismatch. In: EI: electronic imaging
39. Qian Y, Dong J, Wang W, Tan T (2015) Deep learning for steganalysis via convolutional neural networks. In: ISAndamp;t/SPIE electronic imaging, pp 94,090j–94,090j
40. Qian Y, Dong J, Wang W, Tan T (2016) Learning and transferring representations for image steganalysis using convolutional neural network. In: 2016 IEEE international conference on Image processing (ICIP). IEEE, Piscataway, pp 2752–2756
41. Qian Y, Dong J, Wang W, Tan T (2016) Learning representations for steganalysis from regularized cnn model with auxiliary tasks. In: Proceedings of the 2015 international conference on communications, signal processing, and systems. Springer, Berlin, pp 629–637

42. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
43. Sedighi V, Cogranne R, Fridrich J (2016) Content-adaptive steganography by minimizing statistical detectability. IEEE Trans Inf Forensics Secur 11(2):221–234
44. Shi YQ, Chen C, Chen W (2007) A markov process based approach to effective attacking jpeg steganography. In: Information hiding. Springer, Berlin, pp 249–264
45. Shi YQ, Sutthiwan P, Chen L (2013) Textural features for steganalysis. In: Information hiding. Springer, Berlin, pp 63–77
46. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
47. Tan S, Li B (2014) Stacked convolutional auto-encoders for steganalysis of digital images. In: Signal and information processing association annual summit and conference (APSIPA), 2014 asia-pacific. IEEE, Piscataway, pp 1–4
48. Tang W, Li H, Luo W, Huang J (2014) Adaptive steganalysis against wow embedding algorithm. In: Proceedings of the 2nd ACM workshop on information hiding and multimedia security, pp 91–96
49. Wu S, Zhong SH, Liu Y (2016) Steganalysis via deep residual network. In: 2016 IEEE 22nd international conference on Parallel and distributed systems (ICPADS). IEEE, Piscataway, pp 1233–1236
50. Xia Z, Wang X, Sun X, Wang B (2014) Steganalysis of least significant bit matching using multi-order differences. Security and Communication Networks 7(8):1283–1291
51. Xia Z, Wang X, Sun X, Liu Q, Xiong N (2016) Steganalysis of lsb matching using differences between nonadjacent pixels. Multimed Tools Appl 75(4):1947–1962
52. Xu G, Wu HZ, Shi YQ (2016) Ensemble of cnns for steganalysis: an empirical study. In: Proceedings of the 4th ACM workshop on information hiding and multimedia security. ACM, New York, pp 103–107
53. Xu G, Wu HZ, Shi YQ (2016) Structural design of convolutional neural networks for steganalysis. IEEE Signal Process Lett 23(5):708–712
54. Xuan G, Shi YQ, Gao J, Zou D, Yang C, Zhang Z, Chai P, Chen C, Chen W (2005) Steganalysis based on multiple features formed by statistical moments of wavelet characteristic functions. In: Information hiding. Springer, Berlin, pp 262–277
55. Yuan C, Xia Z, Sun X (2017) Coverless image steganography based on sift and bof. Journal of Internet Technology 18(2):435–442

Yinlong Qian is currently a PhD student in the University of Science and Technology of China. He is also working as an intern at the Center for Research on Intelligent Perception and Computing, National Laboratory of Pattern Recognition, Institute of Automation of the Chinese Academy of Sciences. He received his B.S. degree from University of Science and Technology of China in 2011. His research interests include steganalysis, steganography, and deep learning.

Jing Dong received her B.Sc in Electronic Information Science and Technology from Central South University in 2005 and her Ph.D in Pattern Recognition from Graduate University of Chinese Academy of Sciences. Since July 2010, Dr. Dong has joined NLPR where she is currently Associate Professor. Her research interests are towards Pattern Recognition, Image Processing and Digital Image Forensics including digital watermarking, steganalysis and tampering detection. She has published several academic papers and she is a member of CCF,CAAI, IEEE, IEEE Computer Science Society, Signal Society and Communication Society. She also has served as the deputy general of Chinese Association for Artificial Intelligence and an IEEE Volunteer leader in R10 and in Beijing Section from many aspects of academic activities.

Wei Wang received his B.E. in Computer Science and Technology from North China Electric Power University in 2007. Since July 2012, Dr. Wang has joined the National Laboratory of Pattern Recognition (NLPR). He is currently an assistant professor. His research interests are Pattern Recognition, Image Processing and Digital Image Forensics including watermarking, steganalysis and tampering detection.

Tieniu Tan received his BSc in electronic engineering from Xi'an Jiaotong University, China in 1984, and his MSc and PhD degrees in electronic engineering from Imperial College London, UK in 1986 and 1989 respectively. He is currently Director of the Center for Research on Intelligent Perception and Computing at the Institute of Automation. He has published more than 450 research papers in refereed international journals and conferences in the areas of image processing, computer vision and pattern recognition, and has authored or edited 11 books. He holds more than 70 patents. His current research interests include biometrics, image and video understanding, and information forensics and security. He was the Deputy Secretary-General of the Chinese Academy of Sciences till 2015. TAN is a Member (Academician) of the Chinese Academy of Sciences, Fellow of The World Academy of Sciences for the advancement of science in developing countries (TWAS), International Fellow of the UK Royal Academy of Engineering, and Fellow of the IEEE and the IAPR (the International Association of Pattern Recognition). He currently serves as Deputy President of the Chinese Association for Artificial Intelligence, and is a past President of the IEEE Biometrics Council and a past Vice President of the IAPR. He was the Founding Chair of International Conference on Biometrics (ICB), the IEEE International Workshop on Visual Surveillance, Asian Conference on Pattern Recognition (ACPR) and Chinese Conference on Pattern Recognition (CCPR). He is or has served as Associate Editor or member of editorial boards of many leading international journals including IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on Automation Science and Engineering, IEEE Transactions on Information Forensics and Security, Pattern Recognition, etc. He is Editor-in-Chief of the International Journal of Automation and Computing.